Тема 6. Паросочетания и покрытия

6.1. Независимые и покрывающие множества

Определение. Говорят, что вершина *покрывает* инцидентные ей ребра, а ребро *покрывает* инцидентные ему вершины.

Определение. Множество вершин, покрывающих все ребра, называется *вершинным покрытием*. Наименьшее число вершин, образующих вершинное покрытие, называется *числом вершинного покрытия* и обозначается α_0.

Примеры.

Для полного графа $\alpha_0(K_p) = p - 1$.

Для полного двудольного графа $\alpha_0(K(p_1, p_2) = \min(p_1, p_2)$.

Для пустого графа $\alpha_0(\overline{K}_p) = 0$.

Для несвязного графа $\alpha_0(G_1 \cap G_2) = \alpha_0(G_1) + \alpha_0(G_2)$.

Определение. Множество ребер, покрывающих все вершины, называется *реберным покрытием*. Наименьшее число ребер, образующих реберное покрытие, называется *числом реберного покрытия* и обозначается α_1.

Примеры.

Для полного графа с четным числом вершин $\alpha_1(K_p) = p/2$, для полного графа с нечетным числом вершин $\alpha_1(K_p) = (p + 1)/2$.

Для полного двудольного графа $\alpha_1(K(p_1, p_2) = \max(p_1, p_2)$.

Для графа с изолированными вершинами α_1 не определено.

Для четного цикла $\alpha_1(C_p) = p/2$, для нечетного цикла $\alpha_1(C_p) = (p + 1)/2$.

Для несвязного графа $\alpha_1(G_1 \cap G_2) = \alpha_1(G_1) \cap \alpha_1(G_2)$.

Определение. Множество вершин называется *независимым* (*внутренне устойчивым*), если никакие две из них не смежны. Наибольшее число вершин, образующих независимое множество, называется *вершинным числом независимости* и обозначается β_0.

Примеры.

Для полного графа $\beta_0(K_p) = 1$.

Для полного двудольного графа $\beta_0(K(p_1, p_2) = \max(p_1, p_2)$.

Для пустого графа $\beta_0(\overline{K}_p) = p$.

Для несвязного графа $\beta_0(G_1 \cap G_2) = \beta_0(G_1) + \beta_0(G_2)$.

Определение. Множество ребер называется *независимым* (*паросочетанием*), если никакие два из них не смежны. Наибольшее число ребер, образующих независимое множество, называется *реберным числом независимости* и обозначается β_1.

Для полного графа с четным числом вершин $\beta_1(K_p) = p/2$, для полного графа с нечетным числом вершин $\beta_1(K_p) = (p - 1)/2$.

Для полного двудольного графа $\beta_1(K(p_1, p_2) = \min(p_1, p_2)$.

Для четного цикла $\beta_1(C_p) = p/2$, для нечетного цикла $\beta_1(C_p) = (p - 1)/2$.

Для несвязного графа $\beta_1(G_1 \cap G_2) = \beta_1(G_1) + \beta_1(G_2)$.

Приведенные примеры наводят на мысль, что числа независимости и покрытий связаны друг с другом и с количеством вершин графа.

Теорема о числах независимости и покрытий. *Для любого нетривиального связного графа*

$$\alpha_0 + \beta_0 = \alpha_1 + \beta_1 = p.$$

Доказательство. Покажем, что имеют место четыре неравенства, из которых следуют два требуемых равенства.

– $\alpha_0 + \beta_0 \geq p$: пусть M_0 – наименьшее вершинное покрытие, т. е. $|M_0| = \alpha_0$. Тогда $V \setminus M_0$ – независимое множество, т. к. если бы в $V \setminus M_0$ были две смежные вершины, то соединяющее их ребро не было бы покрыто вершинами из M_0. Отсюда $|V \setminus M_0| \leq \beta_0$, следовательно,

$$p = |M_0| + |V \setminus M_0| \leq \alpha_0 + \beta_0.$$

– $\alpha_0 + \beta_0 \leq p$: пусть N_0 – наибольшее независимое множество вершин, т. е. $|N_0| = \beta_0$. Тогда $V \setminus N_0$ – вершинное покрытие, т. к. нет ребер, инцидентных только вершинам из N_0, значит, любое ребро инцидентно по крайней мере одной вершине из $V \setminus N_0$. Отсюда $|V \setminus N_0| \geq \alpha_0$, следовательно,

$$p = |N_0| + |V \setminus N_0| \geq \beta_0 + \alpha_0.$$

– $\alpha_1 + \beta_1 \geq p$: пусть M_1 – наименьшее реберное покрытие, т. е. $|M_1| = \alpha_1$. Множество M_1 не содержит цепей длиной больше 2, т. к. если в M_1 есть цепь длины 3, то среднее ребро этой цепи можно удалить из M_1, и множество останется покрытием, что противоречит тому, что M_1 – наименьшее покрытие. Следовательно, M_1 состоит из *звезд* – графов с диаметром, не большим 2. Пусть этих звезд m, и n_i – число ребер в i-й звезде. Заметим, что звезда из n_i ребер покрывает $n_i + 1$ вершину (т. к. звезда является деревом). Возьмем по одному ребру из каждой звезды и составим из них множество X. Тогда $|X| = m$, множество X независимое, т. е. $|X| \leq \beta_1$. Следовательно

$$p = \sum_{i=1}^{m}(n_i + 1) = \sum_{i=1}^{m} n_i + m = |M_1| + m = |M_1| + |X| \leq \alpha_1 + \beta_1.$$

– $\alpha_1 + \beta_1 \leq p$: пусть N_1 – наибольшее независимое множество ребер, т. е. $N_1 = \beta_1$. Построим реберное покрытие Y следующим образом. Множество N_1 покрывает $2|N_1|$ вершин. Добавим по одному ребру, инцидентному непокрытым $p - 2|N_1|$ вершинам, таких ребер $p - 2|N_1|$. Множество Y – реберное покрытие, т. е. $|Y| \leq \alpha_1$, и $|Y| - |N_1| + p - 2|N_1| = p - |N_1|$, следовательно

$$p = p - |N_1| + |N_1| = |Y| + |N_1| \geq \alpha_1 + \beta_1.$$

Теорема доказана.

Замечание. Условия связности и нетривиальности гарантируют, что все четыре инварианта определены. Однако, это условие является достаточным, но не необходимым.

Например, для графа $K_p \cup K_p$ заключение теоремы остается справедливым, хотя условие не выполнено.

6.2. Независимые множества вершин и вершинные покрытия

6.2.1. Максимальные независимые множества

Задача отыскания наибольшего независимого множества вершин (и, тем самым, определение вершинного числа независимости β_0 принадлежит к числу трудоемких, т. е. требующих перебора.

Определение. Независимое множество вершин называется *максимальным*, если к нему нельзя добывить ни одной вершины так, чтобы оно осталось независимым.

Пример (задача о восьми ферзях. Расставить на шахматной доске восемь ферзей так, чтобы они не били друг друга.

Очевидно, что для нахождения β_0 требуется найти максимальное независимое подмножество наибольшей мощности, т. е. *наибольшее* независимое подмножество.

Поиск с возвратами (backtracking). Идея поиска с возвратами состоит в следующем. Находясь в некоторой ситуации, пробуем изменить ее, чтобы найти решение. Если изменение не привело к успеху, то возвращаемся в исходную ситуацию и пробуем изменить ее другим образом, и так до тех пор, пока не будут исчерпаны все возможности. Поиск с возвратами для отыскания наибольшего независимого множества вершин может быть реализован с помощью следующей рекурсивной процедуры.

Введем обозначения: S – текущее независимое множество вершин, T – вершины графа, которые можно добавить в независимое множество, m – наибольшая мощность независимого множества вершин.

Шаг 1. Полагаем $S = \emptyset$, $T = V$, $m = 0$.

Шаг 2. Если $T \neq \emptyset$, то выбираем очередную вершину $v \in T$, добавляем ее в множество S и повторяем шаги 2–3 при $S = S \cup \{v\}$, $T = T \setminus \Gamma(v)$.

Шаг 3. Если $|S| > m$, то полагаем $m = |S|$ и запоминаем S как наибольшее независимое множество вершин.

Улучшение перебора. Рассмотрим нерекурсивный алгоритм, строящий все максимальные независимые множества. Идея перебора состоит в том, что, начиная с пустого множества, мы добавляем в множество вершины так, чтобы оно оставалось независимым.

Пусть S_k – уже полученное множество из k вершин, Q_k – множество вершин, которые можно добавить к S_k, т. е. $S_k \cap \Gamma(Q_k) = \emptyset$. Среди вершин Q_k будем различать те, которые уже использовались для расширения S_k (обозначим их множество через Q_k^-) и те, которые еще не использовались (обозначим их множество через Q_k^+). Тогда общая схема нерекурсивной реализации поиска с возвратами будет состоять из следующих шагов:

– шаг вперед (от k к $k+1$) состоит в выборе вершины $v \in Q_k^+$, при этом:

$$S_{k+1} = S_k \cup \{v\};$$
$$Q_{k+1}^- = Q_k^- \setminus \Gamma(v);$$
$$Q_{k+1}^+ = Q_k^+ \setminus \{\Gamma(v) \cup v\};$$

– шаг назад (от $k+1$ к k) состоит в удалении вершины v, при этом:

$$S_k = S_{k+1} \setminus \{v\};$$
$$Q_k^- = Q_k^- \cup \{v\};$$
$$Q_k^+ = Q_k^+ \setminus \{v\}.$$

Если S_k – максимальное, то $Q_k^+ = \emptyset$. Если $Q_k^- \neq \emptyset$, то S_k было расширено раньше и не является максимальным. Таким образом, проверка максимальности сводится к проверке условия $Q_k^+ = Q_k^- = \emptyset$.

Перебор можно улучшить, если заметить следующее. Пусть $v \in Q_k^-$ и $\Gamma(v) \cap Q_k^+ = \emptyset$. Тогда вершину v не удастся удалить из Q_k^-, так как из Q_k^- удаляются лишь вершины, смежные с вершинами из Q_k^+. Таким образо, существование в Q_k^- вершины, не смежной ни с одной вершиной из Q_k^+, является достаточным условием для возврата.

Алгоритм построения максимальных независимых множеств вершин.

Шаг 1. Полагаем $k = 0$, $S_0 = \emptyset$, $Q_0^- = \emptyset$, $Q_0^+ = V$.

Шаг 2 (шаг вперед). Выбираем очередную вершину $v \in Q_k^+$. Полагаем $S_{k+1} = S_k \cup \{v\}$, $Q_{k+1}^- = Q_k^- \setminus \Gamma(v)$, $Q_{k+1}^+ = Q_k^+ \setminus \{\Gamma(v) \cup v\}$. Увеличиваем k, т. е. полагаем $k = k+1$.

Шаг 3. Если $Q_k^+ = Q_k^- = \emptyset$, то подмножество S_k – максимальное, записываем его в список максимальных подмножеств и идем на шаг 5.

Шаг 4. Если для всех вершин $u \in Q_k^-$ верно $\Gamma(u) \cap Q_k^+ \neq \emptyset$, идем на шаг 2.

Шаг 5 (шаг назад). Удалаяем из множества S_k последний добавленный элемент v. Полагаем $k = k-1$, $S_k = S_{k+1} \setminus \{v\}$, $Q_k^- = Q_k^- \cup \{v\}$, $Q_k^+ = Q_k^+ \setminus \{v\}$. Если $k = 0$ и $Q_k^+ = \emptyset$, перебор завершен, конец. Иначе идем на шаг 3.

6.2.2. Вершинные покрытия

Определение. Вершинное покрытие называется *кратчайшим*, если при удалении любой вершины оно перестает быть покрытием.

Задача отыскания кратчайшего вершинного покрытия (и, тем самым, определение вершинного числа независимости α_0 принадлежит к числу трудоемких, т. е. требующих перебора.

Пусть вершинам приписаны некоторые цены.

Определение. Покрытие минимальной цены называется *минимальным*.

Сформулировать эту задачу можно еще и так: построим булеву матрицу, строкам которой припишем вершины графа, а столбцам – ребра. Единицы поставим в том случае, если вершина, приписанная строке, покрывает ребро, приписанное столбцу. Каждой строчке

припишем цену (в частном случае все цены равны 1). *Покрытием* назовем множество строк, содержащих в совокупности по крайней мере одну единицу в каждом столбце.

Задача поиска покрытия с наименьшей ценой относится к числу переборных и обсуждалась в курсе дискретной математики. Она носит название *задачи о наименьшем покрытии.*

Пусть $G(V, E)$ – граф, I – его матрица инцидентности, c_j – вес вершины v_j. Тогда в терминах линейного программирования задача о наименьшем вершинном покрытии формулируется следующим образом.

$$\sum_{j=1}^{p} c_j \xi_j \to \min;$$

$$\sum_{j=1}^{p} I_{jk} \xi_j \geq 1, \quad \forall k = 1, \ldots, q;$$

$$\xi_j \in \{0, 1\}.$$

Здесь $\xi_j = 1$, если и только если вершина v_j входит в покрытие.

6.2.3. Доминирующие множества

Определение. Для графа (орграфа) $G(V, E)$ множество вершин $S \in V$ называется *доминирующим*, если $S \cup \Gamma(S) = V$, т. е. для любой вершины $v \in V$ либо $v \in S$, либо $\exists u \in S : (u, v) \in E$.

Определение. Доминирующее множество называется *минимальным*, если никакие его подмножество не является доминирующим.

Определение. Доминирующее множество называется *наименьшим*, если число элементов в нем минимально.

Пример (задача о пяти ферзях). Расставить на шахматной доске пять ферзей так, чтобы они били всю доску.

Доминирование тасно связано с вершинной независимостью.

Теорема. *Независимое множество вершин является максимальным тогда и только тогда, когда оно является доминирующим.*

Доказательство.

Необходимость (от противного). Пусть множество вершин S – максимальное независимое. Допустим, что оно не доминирующее. Тогда существует вершина v, находящаяся на расстоянии больше 1 от всех вершин множества S. Эту вершину можно добавить к S с сохранением независимости, что противоречит тому, что множество S максимальное.

Достаточность (от противного). Пусть множество вершин S – независимое доминирующее. Допустим, что оно не максимальное. Тогда существует вершина v, не смежная ни с одной вершиной из множества S, т. е. находящаяся на расстоянии больше единицы от всех вершин множества S. Это противоречит тому, что множество S доминирующее.

Теорема доказана.

Пусть каждой вершине сопоставлена некоторая цена (в частном случае все цены равны 1). Требуется найти доминирующее множество с наименьшей суммарной ценой вершин. Эта задача является частным случаем *задачи о наименьшем покрытии*.

6.3. Паросочетания и реберные покрытия

Определение. Реберное покрытие называется *кратчайшим*, если при удалении любого ребра оно перестает быть покрытием.

Пусть ребрам приписаны некоторые веса.

Определение. Покрытие минимального веса называется *минимальным*.

Теорема о паросочетании и покрытии. Решение задачи о паросочетании максимальной мощности дает решение задачи о покрытии минимальной мощности, и наоборот.

Доказательство (конструктивное).

От паросочетания к покрытию. Пусть M – произвольное паросочетание, тогда выберем произвольную вершину v, не инцидентную ни одному ребру паросочетания, и присоединим к M любое ребро, инцидентное вершине v. Повторяя процедуру до покрытия всех вершин, получим покрытие C'.

От покрытия к паросочетанию. Пусть C – произвольное покрытие, тогда выберем произвольную вершину v, инцидентную более чем одному ребру покрытия, и исключим из C любое ребро, инцидентное вершине v. Повторяя процедуру до тех пор, пока есть такие вершины, получим паросочетание M'.

Докажем теперь, что если M – паросочетание максимальной мощности, то C' – покрытие минимальной мощности. Паросочетание M покрывает $2M$ вершин. Рассмотрим непокрытые вершины, их $p-2|M|$. Все они попарно не смежны, т. к. иначе, если бы среди них были две смежные вершины, то соединяющее их ребро можно было бы добавить к паросочетанию M, что противоречит тому, что данное паросочетание имеет максимальную мощность. Значит, чтобы покрыть оставшиеся $p-2|M|$ вершин, нужно $p-2|M|$ ребер. Отсюда $|C'| = |M| + p - 2|M| = p - |M|$, т. е. $|M| + |C'| = p$. Значит, если C' имеет не минимальную мощность, то M имеет не максимальную мощность.

Докажем теперь, что если C – покрытие минимальной мощности, то M' – паросочетание максимальной мощности. Если бы покрытие C было паросочетанием, оно бы покрывало $2|C|$ вершин, но в графе p вершин, значит, в покрытии $2|C| - p$ лишних ребер (мы удаляем ребра, смежные с другими ребрами). Отсюда $|M'| = |C| - (2|C| - p) = p - |C|$, т. е. $|C| + |M'| = p$. Значит, если M' имеет не максимальную мощность, то C имеет не минимальную мощность.

Теорема доказана.

Задача поиска кратчайшего реберного покрытия – частный случай *задачи о наименьшем покрытии*, т. е. задачи поиска паросочетаний и реберных покрытий являются вычислительно сложными.

Пусть $G(V, E)$ – граф, I – его матрица инцидентности, c_j – вес ребра e_j. Тогда в терминах линейного программирования задача о наименьшем реберном покрытии формулируется следующим образом.

$$\sum_{j=1}^{q} c_j \xi_j \to \min;$$
$$\sum_{j=1}^{q} I_{kj} \xi_j \geq 1, \quad \forall k = 1, \ldots, p;$$
$$\xi_j \in \{0, 1\}.$$

Здесь $\xi_j = 1$, если и только если ребро e_j входит в покрытие.

Задача о наибольшем паросочетании формулируется следующим образом.

$$\sum_{j=1}^{q} c_j \xi_j \to \max;$$
$$\sum_{j=1}^{q} I_{kj} \xi_j \leq 1, \quad \forall k = 1, \ldots, p;$$
$$\xi_j \in \{0, 1\}.$$

Здесь $\xi_j = 1$, если и только если ребро e_j входит в паросочетание.

6.3. Паросочетание максимальной мощности

6.3.1. Чередующиеся цепи и деревья

Пусть задан граф $G(V, E)$ и паросочетание M в нем.

Определение. *Чередующаяся цепь* – это простая цепь, ребра которой попеременно принадлежат и не принадлежит паросочетанию.

Определение. Вершина называется *открытой*, если она не инцидентна ни одному из ребер паросочетания.

Определение. Чередующаяся цепь называется *увеличивающей*, если ее первая и последняя вершины являются открытыми.

Пример.

Из определения следует, что в увеличивающей цепи нечетное число ребер, причем ребер, не входящих в паросочетание, на одно больше, чем входящих.

Основная теорема, относящаяся к паросочетаниям, может быть сформулирована так.

Теорема о паросочетании максимальной мощности. *Паросочетание M является паросочетанием максимальной мощности тогда и только тогда, когда в $G(V, E)$ не существует никакой увеличивающей цепи.*

Доказательство.

Необходимость (от противного). Пусть M является паросочетанием максимальной мощности и существует увеличивающая цепь P. Никакое ребро из множества $M \setminus (P \cap M)$ не смежно ни с каким ребром из P, т. к. концы цепи по определению открыты, т.

е. не инцидентны ребрам из M, а остальные вершины инцидентна ребрам из $P \cap M$. Следовательно, множество ребер $M' = (M \setminus (P \cap M)) \cup (P \setminus (P \cap M)) = (M \cup P) \setminus (M \cap P)$, полученные в результате замены в M ребер, входящих в $P \cap M$, на ребра из P, не входящие в $P \cap M$, также является паросочетанием. Но т. к. оба концевых ребра цепи P не лежат в M, то $|P \setminus (P \cap M)| > |P \cap M|$, а значит, $|M'| > |M|$ и M не является паросочетанием максимальной мощности.

Достаточность. Пусть M – паросочетание, относительно которого в графе нет ни одной увеличивающей цепи, а M^* – паросочетание максимальной мощности. Построим подграф G', образованный ребрами из $(M \cup M^*) \setminus (M \cap M^*)$ вместе с соответствующими этим ребрам вершинами. Все вершины из этого подграфа имеют степень, не большую 2, т. к. иначе два или более ребер из M или M^* были бы смежными, что противоречит определению паросочетания. Таким образом, компоненты графа G' – это либо простые цепи, либо простые циклы. Никакие два смежных ребра этих цепей и циклов не могут принадлежать одному паросочетанию (по определению паросочетания), значит, ребра в цепях и циклах чередуются: одно принадлежит M, следующее M^*, и т. д. Рассмотрим все виды таких цепей и циклов:

– цепь нечетной длины, первое и последнее ребро которой принадлежат M^*, не может существовать, т. к. она является увеличивающей относительно M, что противоречит условию теоремы;

– цепь нечетной длины, первое и последнее ребро которой принадлежат M, не может существовать, т. к. она является увеличивающей относительно M^*, а M^* имеет максимальную мощность, следовательно, в графе нет увеличивающих относительно M^* цепей;

– цепь четной длины может существовать, причем число ребер в ней, принадлежащих M, равно числу ребер, принадлежащих M^*;

– цикл нечетной длины не может существовать, т. к. содержит два подряд ребра из M или M^*, что противоречит определению паросочетания;

– цикл четной длины может существовать, причем число ребер в нем, принадлежащих M, равно числу ребер, принадлежащих M^*.

Сдедовательно, компонентами графа G' могут быть лишь цепи и циклы четной длины, и в каждой их них число ребер, принадлежащих M, равно числу ребер, принадлежащих M^*. Значит, это верно и для всего графа G': $q_M = q_{M^*}$, где q_M – число ребер в G', принадлежащих M, а q_{M^*} – число ребер в G', принадлежащих M^*. Отсюда $|M| = q_M + |M \cap M^*| = q_{M^*} + |M \cap M^*| = |M^*|$, т. е. M – паросочетание максимальной мощности.

Теорема доказана.

Определение. *Чередующимся деревом* относительно данного паросочетания M называется дерево T, для которого:

– одна вершина, называемая *корнем* дерева T, является открытой;

– все начинающиеся в корне цепи являются чередующимися;

– все *максимальные* (начинающиеся в корне и заканчивающиеся в *листе* – вершине степени 1) цепи содержат четное число ребер.

Пример.

Разобьем все вершины чередующегося дерева на два класса: *внешних* и *внутренних* вершин. Корень дерева отнесем к внешним вершинам. Вершины любой цепи, начинающейся в корне, отнесем поочередно к внутренним и внешним вершинам. Таким образом, если длина простой цепи, соединяющей корень с вершиной v, нечетная, то v – внутренняя вершина, иначе v – внешняя вершина. Все листья будут внешними вершинами. Степень любой внутренней вершины равна 2, а степень внешней вершины может быть любым положительным числом.

Пример.

Определение. *Увеличивающее дерево* по отношению к паросочетанию M – это чередующееся дерево по отношению к этому паросочетанию, удовлетворяющее условию: существует ребро от какой-нибудь внешней вершины дерева u до некоторой открытой вершины w, не принадлежащей дереву.

Пример.

Если в графе есть увеличивающее дерево, то елинственная цепь, идущая от корня дерева к вершине u и затем к открытой вершине w является увеличивающей цепью. Тогда, согласно теореме о паросочетании максимальной мощности, мощность паросочетания M может быть увеличена, как показано в доказательстве достаточности.

6.3.2. Цветки

Определение. *Цветком* по отношению к паросочетанию M называется чередующаяся цепь нечетной длины, первая и последняя вершины которой совпадают, а первое и последнее ребро не принадлежат M.

Пример.

При поиске паросочетания максимальной мощности цветки *срезают* для того, чтобы получить более простой новый граф. Эта операция состоит в стягивании цикла в *псевдовершину*. В упрощенном графе, полученном после срезания цветка, псевдовершины могут в свою очередь образовывать новый цветок, который опять срезается. Последний цветок, не содержащийся ни в каких других цветках, называется *крайним цветком*.

Определение. *Цветущее дерево* – это чередующееся дерево относительно данного паросочетания, для которого существует ребро, соединяющее две внешние вершины дерева.

Пример.

Пусть v – корень дерева, u, u' – внешние вершины, причем $(u, u') \in E$. Тогда существует замкнутый маршрут, состоящий из цепи дерева $< v, u >$, ребра (u, u') и цепи $< u', v >$. В этом маршруте есть простой цикл нечетной длины C_b, содержащий внешнюю вершину b, которая принадлежит обеим цепям: $< v, u >$ и $< u', v >$, который и является цветком. Этот цикл стягивается в псевдовершину, которая в дереве является внешней. Остальные

вершины в цикле могут быть как внешними, так и внутренними, это зависит от направления обхода цикла.

Теорема о цветке. *Если B – цветок с множеством вершин X_B, а x – некоторая вершина из X_B, то в подграфе, образованном вершинами X_B, существует паросочетание максимальной мощности, для которого вершина x будет открытой.*

Доказательство. Пусть $x_1x_2 \dots x_\alpha x_1$ – цикл с нечетным числом вершин, образующий цветок B. Тогда возможны две ситуации:

– $x = x_i$ для некоторого $i \in \{1, 2, \dots \alpha\}$;

– $x \in x_i$ для некоторой псевдовершины x_i, соответствующей ранее срезанному цветку B_i.

В первом случае вершина x остается открытой, а остальные вершины цикла (их осталось четное число) разбиваются на пары в порядке прохождения цикла, образуя паросочетание. Если одна из этих вершин, например, x_j, соответствует ранее срезанному цветку B_j, то одна из вершин этого цветка инцидентна ребру паросочетания, а остальные вершины этого цветка таким же образом разбиваются на пары. Этот процесс продолжается, пока не будут разбиты на пары все вершины, отличные от x.

Во втором случае вершина x остается открытой, а остальные вершины цикла аналогично разбиваются на пары. В результате относительно цветка B_i мы получаем ту же ситуацию, которая была в самом начале для цветка B.

Теорема доказана.

Пример.

6.3.3. Венгерские деревья

Определение. *Венгерское дерево* – это такое чередующееся дерево, что каждое ребро графа, инцидентное внешней вершине, инцидентно также и внутренней вершине (т. е. нет ребер, соединяющих две внешние вершины).

Пример.

Теорема о венгерском дереве. *Пусть H – венгерское дерево в графе $G(V, E)$, и $G_0(V_0, E_0)$ – подграф графа $G(V, E)$, полученный удалением из $G(V, E)$ множества вершин V_H, принадлежащих дереву H. Если M_H – паросочетание в дереве H и M_0^* – паросочетание максимальной мощности в подграфе $G_0(V_0, E_0)$, то $M_H \cup M_0^*$ является паросочетанием максимальной мощности в $G(V, E)$.*

Доказательство. Пусть V_H – множество вершин дерева H. Разобьем множество E ребер графа G на три подмножества:

$$E_H = \{(u, v) | (u, v) \in E,\ u, v \in V_H\};$$
$$E_{0H} = \{(u, v) | (u, v) \in E,\ u \in V_H,\ v \in V \setminus V_H\};$$
$$E_0 = \{(u, v) | (u, v) \in E,\ u, v \in V \setminus V_H\}.$$

Если S – произвольное паросочетание в графе $G(V, E)$, то его можно разбить на аналогичные подмножества:

$$S = S_H \cup S_{0H} \cup S_0;$$
$$S_H = S \cap E_H, \quad S_{0H} = S \cap E_{0H}, \quad S_0 = S \cap E_0.$$

По определению паросочетания M_0^*

$$|M_0^*| \geq |S_0|.$$

Рассмотрим теперь граф G', состоящий из ребер $E_H \cup E_{0H}$ и их концевых вершин. По отношению к S_H все вершины из $V_0 = V \setminus V_H$, являющиеся концевыми для ребер из E_{0H}, открытые. Чередующая цепь, начинающаяся в открытой вершине $v \in V_0$ (или в корне дерева H, который также является открытой вершиной), тогда и только тогда может быть увеличивающей, когда она оканчивается в другой вершине $u \in V_0$. Но чередующаяся цепь должна содержать нечетное число ребер, а это значит, что если первое ребро соединяет открытую вершину с внутренней, то последнее – внешнюю с открытой. Так как по определению дерева H все ребра в E_{0H} соединяют внутренниме вершины дерева с вершинами из V_0, то в графе G' нет никакой увеличивающей цепи и, следовательно, M_H является паросочетанием максимальной мощности в графе G', т. е.

$$|M_H| \geq |S_H| + |S_{0H}|.$$

Окончательно получаем

$$|M_H \cup M_0^*| = |M_H| + |M_0^*| \geq |S_H| + |S_{0H}| + |S_0| = |S|.$$

Таким образом, $M_H \cup M_0^*$ – паросочетание максимальной мощности в графе G.

Теорема доказана.

6.3.4. Поиск паросочетания максимальной мощности

Алгоритм поиска паросочетания максимальной мощности основан на рассмотренных теоремах и состоит в следующем. Пусть имеется некоторое начальное паросочетание (допустимо использование пустого паросочетания). В графе строится чередующее дерево, в результате возможны три исхода:

а) дерево станет увеличивающим;

б) дерево станет цветущим;

в) дерево станет венгерским.

В случае а) число ребер в паросочетании может быть увеличено на единицу, как показано в теореме о паросочетании максимальной мощности. После этого строится новое чередующееся дерево.

В случае б) цветок срезается, и продолжается построение дерева. Порядок, в котором срезаются цветки, важен, так как в конце процедуры в срезанном цветке строится паросочетание, как указано в теореме о цветке, при этом цветок «расцветает» в обратном порядке.

В случае в) вершины венгерского дерева и инцидентные им ребра, в соответствии с теоремой о венгерском дереве, удаляются из графа, и алгоритм применяется к оставшемуся подграфу.

Алгоритм построения чередующегося дерева.

Начало. Задан граф $G(V, E)$ и паросочетание M в этом графе.

Шаг 1. Выбираем любую открытую вершину v. Назначаем ее корнем и помечаем как внешнюю. Все остальные вершины считаем напомеченными, все ребра – неокрашенными.

Шаг 2. Выбираем любое неокрашенное ребро (x, y), инцидентное внешней вершине x. Если такого ребра нет, идем на шаг 4. Возможны три случая:

а) y – внутренняя вершина дерава, тогда окрашиваем (x, y) в синий цвет (ребро не принадлежит дереву) и идем на шаг 2;

б) y – внешняя вершина дерева, тогда окрашиваем (x, y) в красный цвет (ребро принадлежит дереву) и идем на шаг 3;

в) y – непомеченная вершина, тогда окрашиваем (x, y) в красный цвет (ребро принадлежит дереву). Если y – открытая вершина, то получено *увеличивающее дерево*; выделяем в нем увеличивающую цепь $< v, y >$, состоящую из красных ребер, и идем на конец. Иначе, если вершине y инцидентно ребро $(y, z) \in M$, то окрашиваем (y, z) в красный цвет (ребро принадлежит дереву), помечаем y как внутреннюю, а z – как внешнюю, и идем на шаг 2.

Шаг 3. Ребро (x, y) окрашено в красный цвет и инцедентно двум внешним вершинам, т. е. в дереве имеется цикл нечетной длины, и получено *цветущее дерево* с цветком B. Идем на конец.

Шаг 4. Дальнейшее окрашивание ребер невозможно, получено *венгерское дерево*, состоящее из красных ребер.

Конец.

Алгоритм построения паросочетания максимальной мощности.

Начало. Задан граф G_0 и в нем паросочетание M_0. Полагаем $i = 0$.

Шаг 1. Если в графе G_i есть по крайней мере две открытые вершины, то назначаем одну из них корнем дерева v. Иначе идем на шаг 6.

Шаг 2. Строим чередующееся дерево с корнем в открытой вершине v. Если получено увеличивающее дерево, идем на шаг 3, если цветущее дерево, идем на шаг 4, если венгерское дерево, идем на шаг 5.

Шаг 3. Исключаем из паросочетания M_i ребра, вошедшие в найденную увеличивающую цепь $< v, y >$, и добавляем к нему оставшиеся ребра этой цепи. Отменяем пометки вершин и окраску ребер дерева. Идем на шаг 1.

Шаг 4. Полагаем $i = i + 1$. Найденный нечетный цикл B_i стягиваем в фиктивную вер-

шину b_i, получаем граф G_i. Образуем паросочетание M_i из всех ребер паросочетания M_{i-1}, принадлежащих графу G_i. Вершину b_i полагаем внешней вершиной дерева и продолжаем строить чередующееся дерево – идем на шаг 2.

Шаг 5. Запоминаем венгерское дерево H_i и паросочетание S_i в этом дереве. Полагаем $i = i + 1$. Получаем граф G_i удалением из G_{i-1} всех вершин дерева H_i. Образуем паросочетание M_i из всех ребер паросочетания M_{i-1}, принадлежащих графу G_i. Идем на шаг 1.

Шаг 6. Выделить в последнем графе G_i (и в каждом удаленном венгерском дереве H_j) паросочетание максимальной мощности. Для этого, если в графе есть фиктивная вершина b_k, распускаем крайний цветок B_k, выделяем в нем в нем паросочетание, оставляя открытой вершину, которая покрыта ребром из M_i (S_j), и добавляем это паросочетание к M_i (S_j). Повторяем распускание цветков до тех пор, пока в графе есть фиктивные вершины.

Конец. Объединение паросочетаний M_i (для остатка графа) и S_j (для всех венгерских деревьев) дает паросочетание максимальной мощности M^*.

Пример.

6.4. Паросочетание максимального веса

6.4.1. Постановка задачи

Пусть $G(V, E)$ – граф, I – его матрица инцидентности, c_j – вес ребра e_j. Тогда в терминах линейного программирования задача о наибольшем паросочетании формулируется следующим образом.

$$\sum_{j=1}^{q} c_j \xi_j \to \max;$$

$$\sum_{j=1}^{q} I_{kj} \xi_j \le 1, \quad \forall k = 1, \ldots, p;$$

$$\xi_j \in \{0, 1\}.$$

Здесь $\xi_j = 1$, если и только если ребро e_j входит в паросочетание.

Решение задачи дает некоторая вершина выпуклого многогранника, задаваемого неравенствами $\sum\limits_{j=1}^{q} I_{kj}\xi_j \le 1$. Общие свойства таких многогранников содержатся в следующей теореме.

Теорема (Балински). *Все вершины выпуклого многогранника, задаваемого неравенствами* $\sum\limits_{j=1}^{q} I_{kj}\xi_j \le 1$, *где* I *– матрица инцидентности графа, имеют координаты со значениями 0,* $1/2$ *или 1.*

Случай нецелочисленных значений $\xi_j = 1/2$ отвечает графам, имеющим циклы с нечетным числом ребер. Если, например, граф состоит из единственного цикла из нечетного числа ребер, веса которых равны 1, то решением будет $\xi_1 = \ldots = \xi_q = 1/2$. При этом значение решения будет 2.5, хотя наибольшее паросочетание здесь содержит два ребра.

Если граф содержит четное число вершин, то ограничения-неравенства становятся равенствами (т. к. тогда каждая вершина становится инцидентна ровно одному ребру. Граф с нечетным числом вершин p можно преобразовать в полный граф K_{p+1}, добавив еще одну вершину, и приписав ребрем, не содержащимся в исходном графе, вес, равный $-\infty$.

Определение. Паросочетание, покрывающее все вершины графа, называется *совершенным паросочетанием.*

Задача поиска максимального совершеного паросочетания имеет вид

$$\sum_{j=1}^{q} c_j \xi_j \to \max;$$
$$\sum_{j=1}^{q} I_{kj} \xi_j = 1, \quad \forall k = 1, \ldots, p;$$
$$\xi_j \in \{0, 1\}.$$

Условия целочисленности $\xi_j \in \{0, 1\}$ могут быть исключены для графов, не содержащих циклов нечетной длины, т. е. для *двудольных* графов, которые будут рассмотрены ранее. Для произвольного графа эти условия не являются излишними, однако могут быть заменены следующими условиями.

Для каждого подмножества $V_r \subseteq V$, содержащего нечетное число вершин $2p_r + 1$, вводится ограничение

$$\sum_{e_j \in E_r} \xi_j \leq q_r,$$

где E_r – множество ребер правильного подграфа, образованного вершинами из V_r.

Кроме того, для всех $j = 1, \ldots q$ вводятся ограничения

$$\xi_j \geq 0.$$

Получаем следующую задачу линейного программирования P

$$\sum_{j=1}^{q} c_j \xi_j \to \max;$$
$$\sum_{a_j \in T_k}^{q} \xi_j = 1, \quad \forall\, k = 1, \ldots, p, \quad T_k = \{e_j = (v_i, v_k)\};$$
$$\sum_{e_j \in E_r} \xi_j \leq p_r, \quad \forall\, V_r \subseteq V : |V_r| = 2p_r + 1;$$
$$\xi_j \geq 0, \quad \forall\, j = 1, \ldots q.$$

Двойственной к этой задаче будет следующая задача P^*

$$\sum_{j=1}^{p} \pi_j + \sum_{V_r} p_r \lambda_r \to \min;$$
$$\pi_i + \pi_k + \sum_{V_r \in F_j} \lambda_r \geq c_j, \quad \forall\, j = 1, \ldots, q, \quad e_j = (v_i, v_k), \quad F_j = \{V_r | e_j \in E_r\};$$
$$\lambda_j \geq 0, \quad \forall\, V_r.$$

Таким образом, с каждой вершиной v_i связана переменная π_i, а с каждым подмножеством из нечетного числа вершин S_r – переменная λ_r.

В соответствии с теоремой двойственности линейного программирования, вектор значений переменных ξ_j максимизирует линейную форму в задаче P, а вектор значений переменных π_i, λ_r минимизирует линейную форму в задаче P^* тогда и только тогда, когда:

а) для любого r либо $\lambda_r = 0$, либо в соответствующем ограничении имеет место равенство;

б) для любого j либо $\xi_j = 0$, либо в соответствующем ограничении имеет место равенство.

Ясно, что введенные ограничения выполняются для любого паросочетания, т. е. являются необходимыми. Достаточность мы покажем, описав процедуру, позволяющую строить паросочетание, удовлетворяющее этим условиям. Таким образом будет доказано, что это паросочетание оптимально.

6.4.2. Построение паросочетания максимального веса

Предположим, что мы начинаем с графа $G(V, E)$. Присвоим его вершинам веса π_i, достаточно большие, чтобы ограничения двойственной задачи выполнялись при $\lambda_r = 0$.

Определение. *Специальным остовным подграфом* $G'(V, E')$ графа $G(V, E)$ называется его остовный подграф, содержащий только те ребра $e_j = (v_i, v_k)$, для которых

$$\pi_i + \pi_k + \sum_{V_r \in F_j} \lambda_r = c_j.$$

Если в $G'(V, E')$ можно найти совершенное паросочетание (с помощью алгоритма, изложенного в предыдущем разделе), то это паросочетание – оптимальное. Действительно, вектор $[\pi_i, \lambda_r]$ удовлетворяют ограничениям задачи P^*, условие а) выполнено, т. к. $\lambda_r = 0$, а условие б) выполнено, т. к. в паросочетание входят только ребра из $G'(V, E')$, для которого б) верно по построению.

Может оказаться, что в $G'(V, E')$ совершенное паросочетание найти нельзя, т. е. алгоритм построит венгерское дерево. В этом случае наибольшее паросочетание строится из паросочетания в дереве, при этом корень остается открытым, что означает, что паросочетание покроет не все вершины, т. е. не будет совершенным. В этом случае вектор весов $[\pi_i, \lambda_r]$ модифицируется и строится новый остовный подграф.

Введем следующую индексацию. Первоначальный граф G обозначим через G_0, а его специальный остовный подграф – через G'_1. В этом графе будем строить чередующееся дерево. Когда образуется первый цветок, он срезается, получившийся граф обозначается через G_1, а его специальный остовный подграф через G'_2, и т. д.

Очередная итерация алгоритма такова: пусть после срезания f цветков имеются очередной граф G_{f-1} и его специальный остовный подграф G'_f. В G'_f строится чередующееся дерево до тех пор, пока не произойдет одно из трех событий:

А) дерево станет цветущим;

Б) дерево станет увеличивающим;

В) дерево станет венгерским.

Случай А. Цветок срезается и получается новый граф G_f и его специальный остовный подграф G'_{f+1}. Затем продолжается построение чередующегося дерева.

Случай Б. В этом случае получается паросочетание с большим числом ребер, чем предыдущее. Дерево отбрасывается, и начинается построение нового дерева с корнем в другой открытой вершине. Следует отметить, что при этом некоторые псевдовершины в G'_f, образовавшиеся после срезания более ранних цветков, давших графы $G'_1, \ldots, G'_{f-1}$ могут оказаться помеченными как внутренние в новом дереве.

Случай В. В этом случае вектор весов $[\pi_i, \lambda_r]$ для текущего графа G_{f-1} изменяется так, чтобы получился новый специальный остовный подграф G'_f. Изменения в векторе весов делаются так, что:

а) в новом графе G'_f веса $\lambda_r > 0$ лишь для множеств вершин, соответствующих псевдовершинам из G_{f-1}, и при этом сохраняется равенство в ограничениях для всех ребер или цветков, которые после срезания дали псевдовершины в G_{f-1};

б) текущее чередующееся дерево можно строить дальше: или на нем возникает цветок, или оно станет увеличивающим с помощью новых ребер, входящих в новый граф G'_f.

В результате либо некоторая псевдовершина в текущем чередующемся дереве, помеченная как внутренняя, перестанет быть таковой (цветок можно будет распустить), либо будет показано, что в исходном графе нет совершенного паросочетания.

Рассмотрим изменения вектора весов $[\pi_i, \lambda_r]$. Для ребер $e_j = (v_i, v_k)$ из G_{f-1}, не принадлежащих G'_f, одна концевая вершина которых принадлежит текущему чередующемуся дереву и помечена как внешняя, а другая концевая вершина не принадлежит дереву, находим

$$\Delta_1 = \min_{a_j}[\pi_i + \pi_k - c_j].$$

Для ребер $e_j = (v_i, v_k)$ из G_{f-1}, не принадлежащих G'_f, обе концевые вершины которых принадлежат текущему чередующемуся дереву и помечены как внешние, находим

$$\Delta_2 = \frac{1}{2} \min_{a_j}[\pi_i + \pi_k - c_j].$$

Для множеств S_r вершин из G, стянутых в крайние псевдовершины в текущем чередующемся дереве, помеченные как внутренние, находим

$$\Delta_3 = \frac{1}{2} \min_{S_r} \lambda_r.$$

Берем

$$\Delta = \min\{\Delta_1, \Delta_2, \Delta_3\}.$$

Для каждой вершины v_i из G, являющейся внешней вершиной дерева или содержащейся в псевдовершине дерева, помеченной как внешняя, уменьшаем π_i на величину Δ ($\pi_i = \pi_i - \Delta$).

Для каждой вершины v_i из G, являющейся внутренней вершиной дерева или содержащейся в псевдовершине дерева, помеченной как внутренняя, увеличиваем π_i на величину Δ ($\pi_i = \pi_i + \Delta$).

Для каждого множества S_r вершин из G, образующих крайнюю псевдовершину дерева, помеченную как внешнюю, увеличиваем λ_r на 2δ ($\lambda_r = \lambda_r + 2\Delta$).

Для каждого множества S_r вершин из G, образующих крайнюю псевдовершину дерева, помеченную как внутреннюю, уменьшаем λ_r на 2δ ($\lambda_r = \lambda_r - 2\Delta$).

Безотносительно к значению Δ все ребра из старого графа G'_f, образующие текущее дерево, остаются в новом графе G'_f, т. к. каждое такое ребро соединяет внешнюю и внутреннюю вершины, а веса внешних (внутренних) вершин увеличиваются (уменьшаются) на одну и ту же величину, а значит равенство в ограничениях сохраняется. Равенство сохранится также и для ребер тех цветков, которые были срезаны, образовав псевдовершины графа G_{f-1}. Действительно, если псевдовершина из G_{f-1} была помечена как внешняя в дереве в старом графе G'_f, то для некоторого ребра $e_j = (v_i, v_k)$, входящего в цветок, стянутый в эту псевдовершину, веса π_i и pi_k уменьшатся на Δ, но при этом λ_r для множества вершин S_r, стянутых в эту псевдовершину, увеличится на 2Δ. Если же псевдовершина из G_{f-1} была помечена как внутренняя в дереве в старом графе G'_f, то для некоторого ребра $e_j = (v_i, v_k)$, входящего в цветок, стянутый в эту псевдовершину, веса π_i и pi_k увеличатся на Δ, но при этом λ_r для множества вершин S_r, стянутых в эту псевдовершину, уменьшится на 2Δ. Следовательно, в обоих случаях равенство сохранится.

Предположим теперь, что $\Delta = \Delta_1$ и что ребром, давшим это значение Δ_1, является e^*_j. После изменения вектора весов ребро e^*_j будет удовлетворять ограничениям, причем это ограничение будет равенством, так что это ребро войдет в новый граф G'_f. Таким образом, за счет этого ребра чередующееся дерево (которое останется таковым и в новом графе G'_f) можно либо сделать увеличивающим, либо расширить, в зависимости от того, является ли концевая вершина ребра e^*_j, не принадлежащая дереву, открытой или нет.

Предположим теперь, что $\Delta = \Delta_2$ и что ребром, давшим это значение Δ_2, является e^*_j. После изменения вектора весов ребро e^*_j будет удовлетворять ограничениям, причем это ограничение будет равенством, так что это ребро войдет в новый граф G'_f и приведет к образованию цветка.

Если же $\Delta = \Delta_3$, и множеством вершин, давшим это значение Δ_3, является S^*_r, то соответствующая переменная λ^*_r станет равной нулю. Пусть v^* – внутренняя псевдовершина текущего дерева, соответствующая S^*_r, и B^* – крайний цветок, давший после срезания v^*. Теперь псевдовершина v^* графа G_{f-1} может расцвести, образовав цветок B^*, тем самым будет получен граф G_{f-2}. При этом G_{f-2} не будет тем же самым, из которого получен G_{f-1} после срезания цветка, т. к. срезанный цветок образует внешнюю псевдовершину, а значит, этот цветок не будет тем крайним цветком, который срезали для образования G_{f-1} из G_{f-2}. Пусть G'_{f-1} – специальный остовный подграф нового графа G_{f-2}. Поскольку для всех ребер из B^* в ограничениях имеет место равенство, все эти ребра входят в G'_{f-1}. По-

сле того, как цветок распустился, он разбивает текущее дерево на две компоненты, одна из которых касается цветка в вершине v_1, а другая – в вершине v_2. Эти вершины соединены двумя цепями, содержащими только ребра из цветка, одна из цепей четной длины, другая – нечетной. Выбираем цепь четной длины и добавляем ее к текущему дереву. Если B^* состоит из $2m+1$ ребер, то в нем легко получить паросочетание из m ребер. Получаем его таким образом, чтобы дерево оставалось чередующим. Таким образом, в новом дереве появилась по крайней мере одна новая внешняя вершина, и дерево можно расширять, пока снова не возникнет один из рассмотренных случаев.

Алгоритм заканчивает работу либо когда в очередном графе G'_f будет получено совершенное паросочетание, которое затем преобразуется в паросочетание в исходном графе путем распускания цветков в порядке, обратном их срезанию, и построении в цветках паросочетаний, либо когда в случае B будет получено $\Delta = \infty$, тогда в исходном графе не существует совершенного паросочетания.

Алгоритм построения паросочетания максимального веса.

Начало. Задан граф G_0. Присваиваем его вершинам веса π_i так, чтобы при $\lambda_r = 0$ выполнялось соотношение

$$\pi_i + \pi_k + \sum_{V_r \in F_j} \lambda_r \geq c_j, \quad \forall\, j = 1, \ldots, q, \quad e_j = (v_i, v_k), \quad F_j = \{V_r | e_j \in E_r\}.$$

Полагаем $i = 0$.

Шаг 1. Формируем специальный остовный подграф G'_{i+1} графа G_i, содержащий только те ребра $e_j = (v_i, v_k)$, для которых

$$\pi_i + \pi_k + \sum_{V_r \in F_j} \lambda_r = c_j.$$

Шаг 2. Если в графе G'_{i+1} нет чередующегося дерева, то строим новое дерево T_{i+1} с корнем в некоторой открытой вершине. Если открытых вершин нет, идем на шаг 7. Если дерево уже есть, продолжаем его построение. Если получено увеличивающее дерево, идем на шаг 3, если цветущее дерево, идем на шаг 4, если венгерское дерево, идем на шаг 5.

Шаг 3. Исключаем из паросочетания M_i ребра, вошедшие в найденную увеличивающую цепь $< v, y >$, и добавляем к нему оставшиеся ребра этой цепи. Отменяем пометки вершин и окраску ребер дерева. Идем на шаг 2.

Шаг 4. Полагаем $i = i + 1$. Найденный нечетный цикл B_i стягиваем в фиктивную вершину b_i, получаем граф G_i. Образуем паросочетание M_i из всех ребер паросочетания M_{i-1}, принадлежащих графу G_i. Строим специальный остовный подграф G'_{i+1}. Вершину b_i полагаем внешней вершиной дерева и идем на шаг 2.

Шаг 5. Вычисляем Δ_1, Δ_2, Δ_3 и Δ, как описано выше. Если $\Delta = \infty$, в графе нет совершенного паросочетания, идем на конец. Иначе изменяем векрот весов $[\pi_i, \lambda_r]$. Если $\Delta = \Delta_1$ или $\Delta = \Delta_2$, сохраняем текущее дерево T_{i+1} и идем на шаг 1.

Шаг 6. Положим $i = i - 1$. Распускаем крайний цветок B_k, давший Δ_3, обозначаем полученный граф через G_i, а его специальный остовный подграф через G'_{i+1}. Строим в

цветке паросочетание, добавляем его к M_{i+1} и получаем паросочетание M_i. Добавляем к ребрам дерева T_{i+1} необходимую цепь из цветка, получаем дерево T_i. Идем на шаг 2

Шаг 7. Объединеняем паросочетание M_i с паросочетаниями в цветках, распускаемых в порядке, обратном срезанию, получаем паросочетание максимальной мощности M^*.

Конец.

6.5. Паросочетания в двудольных графах.

6.5.1. Теорема Холла о свадьбах

Теорема о свадьбах, доказанная Филиппом Холлом в 1935 году, отвечает на следующий вопрос, известный под названием *задачи о свадьбах*: рассмотрим некоторое конечное количество юношей, каждый из которых знаком с несколькими девушками; при каких условиях можно женить юношей так, чтобы каждый из них женился на знакомой ему девушке (будем считать, что полигамия не разрешена)?

Пример.

Юноша	Знакомые девушки	Возможное решение
Джон	Мэри, Джулия, Джейн	Джулия
Юджин	Мэри	Мэри
Дэвид	Пэт, Моника, Джулия	Моника
Роберт	Пэт, Джулия	Пэт

Эту задачу можно изложить на языке теории графов. Пусть имеется двудольный граф (долями которого являются множества юношей и девушек); ребра соединяют юношей со знакомыми ему девушками.

Определение. *Совершенным паросочетанием* из V_1 в V_2 в двудольном графе $G(V_1, V_2, E)$ называется взаимно-однозначное соответствие f между вершинами из V_1 и подмножеством вершин из V_2, обладающее тем свойством, что $\forall v \in V_1 : \ (v, f(v)) \in E$.

Тогда задача о свадьбах формулируется следующим образом: при каких условиях в двудольном графе $G(V_1, V_2, E)$ существует совершенное паросочетание из V_1 в V_2?

Используя прежнюю терминологию, можно сформулировать очевидное необходимое условие: любые k юношей должны быть знакомы в совокупности по меньшей мере с k девушками для всех $1 \leq k \leq m$, где m – общее число юношей. Необходимость этого условия сразу следует из того, что если оно не верно для какого-либо множества из k юношей, то мы не сможем женить требуемым образом даже этих k юношей.

Оказывается, это условие является и достаточным.

Теорема Холла. Совершенное паросочетание из V_1 в V_2 в двудольном графе $G(V_1, V_2, E)$ существует тогда и только тогда, когда $\forall A \in V_1 : \ |A| \leq |\Gamma(A)|$.

Доказательство. Необходимость условия очевидна. Докажем достаточность. Воспользуемся методом математической индукции. При $|V_1| = 1$ теорема верна. Предположим, что

утверждение справедливо для $|V_1| = 1, 2, \ldots, m-1$. Пусть $|V_1| = m$. Рассмотрим два возможных случая:

– любые k вершин из V_1 смежны в совокупности по меньшей мере с $k+1$ вершиной из V_2. Тогда, если взять любую вершину из V_1, соединить с любой смежной вершиной из V_2 и удалить это ребро вместе с вершинами из графа, то в V_1 останется $m-1$ вершин, для которых верно условие теоремы, а значит, по ишдуктивному предположению, дли них существует совершенное паросочетание. Добавляя удаленное ребро, получаем совершенное паросочетание в исходном графе.

– пусть теперь имеется k вершин из V_1, смежных в совокупности с k вершинами из V_2. Рассмотрим правильный подграф графа $G(V_1, V_2, E)$, включив в него выбранные вершины из V_1 и смежные с ними вершины из V_2. По индуктивному предположению в этом подграфе существует совершенное паросочетание. Удалим это паросочетание (вершины и ребра) из графа. Остается еще $m-k$ вершин в V_1. Любые l из этих вершин смежны по крайней мере с l из оставшихся вершин в V_2, поскольку иначе вместе с первоначально выбранными k вершинами они смежны менее чем с $k+l$ вершинами, что противоречит условию теоремы. Следовательно, по индуктивному предположению, для оставшихся вершин существует совершенное паросочетание, которое в объединении с удаленными вершинами и ребрами дает совершенное паросочетание для исходного графа.

Теорема доказана.

Пусть имеется множество E и семейство его непустых подмножеств $\mathcal{S} = \{S_1, \ldots, S_m\}$.

Определение. *Трансверсалью* (или *системой различных представителей*) для $\mathcal{S}$ называется подмножество множества E, состоящее из m элементов, включающее по одному элементу из каждого множества S_i.

Легко увидеть связь между этой задачей и задачей о свадьбах: если взять за E множество девушек, а за S_i – подмножество девушек, знакомых юноше b_i, то трансверсалью является множество из m девушек, такое, что каждому юноше соответствует ровно ожна знакомая ему девушка.

Теорема. Пусть E – некоторое множество и $\mathcal{S} = \{S_1, \ldots, S_m\}$ – семейство его непустых подмножеств; тогда $\mathcal{S}$ имеет трансверсаль в том и только в том случае, когда объединение любых k подмножеств содержит по меньшей мере k элементов, $(1 \le k \le m)$.

Доказательство. Необходимость этого условия очевидна. Для доказательства достаточности установим, что если какое-либо из подмножеств содержит более одного элемента, то можно удалить один элемент из этого подмножества, не нарушив условие теоремы. Повторение этой процедуры мы сведем доказательство к случаю, когда каждое подмножество содержит ровно один элемент, и доказательство станет очевидным.

Предположим, что S_1 содержит элементы x и y, удаление каждого из которых нарушает условие теоремы. Тогда существуют подмножества A и B множества $\{2, 3, \ldots, m\}$,

для которых верно

$$\left|\left(\bigcup_{j\in A} S_j\right) \cup (S_1 \setminus \{x\})\right| \leq |A|, \quad \left|\left(\bigcup_{j\in B} S_j\right) \cup (S_1 \setminus \{y\})\right| \leq |B|.$$

Но тогда

$$|A| + |B| + 1 = |A \cup B| + |A \cap B| + 1 \leq \left|\left(\bigcup_{j\in A\cup B} S_j\right) \cup S_1\right| + \left|\bigcup_{j\in A\cap B} S_j\right| \leq_{\text{[по условию]}}$$
$$\leq \left|\left(\bigcup_{j\in A} S_j\right) \cup (S_1 \setminus \{x\})\right| + \left|\left(\bigcup_{j\in B} S_j\right) \cup (S_1 \setminus \{y\})\right| \leq_{\text{[т. к. } |S_1| \geq 2\text{]}} |A| + |B|.$$

Теорема доказана.

Пусть имеется множество E и два семейства его непустых подмножеств $\mathcal{S} = \{S_1, \ldots, S_m\}$ и $\mathcal{T} = \{T_1, \ldots, T_m\}$.

Определение. *Общей трансверсалью* для $\mathcal{S}$ и $\mathcal{T}$ называется подмножество множества E, являющееся трансверсалью для $\mathcal{S}$ и $\mathcal{T}$.

Рассмотрим задачу о составлении расписаний. Пусть E – множество отрезков времени, в которые могут читаться лекции, $\mathcal{S}$ – множества отрезков времени, в которые данные m профессоров желают читать лекции, $\mathcal{T}$ – множества отрезков времени, в которые свободны m лекционных аудиторий. Тогда, найдя общую трансверсаль, мы можем предоставить каждому профессору аудиторию в удобное для него время.

Теорема. Пусть E – некоторое множество и $\mathcal{S} = \{S_1, \ldots, S_m\}$ и $\mathcal{T} = \{T_1, \ldots, T_m\}$ – семейства его непустых подмножеств; тогда $\mathcal{S}$ и $\mathcal{T}$ имеют общую трансверсаль в том и только в том случае, если $\forall A, B \subseteq \{1, \ldots, m\}$

$$\left|\left(\bigcup_{j\in A} S_j\right) \bigcap \left(\bigcup_{j\in B} T_j\right)\right| \geq |A| + |B| - m.$$

6.5.2. Задача о назначениях

Пусть имеется взвешенный двудольный граф $G(V_1, V_2, E)$ с матрицей весов $C =< c_{ij}$ и ставится задача найти совершенное паросочетание с максимальным или минимальным весом. Нахождение паросочетания с минимальным весом известно как *задача о назначениях* и без потери общности часто обсуждается в связи с полными двудольными графами.

Для полного графа $K_{p,p}$ задача может быть решена с помощью методов, предназначенных для исследования потоков в сетях. Добавим к графу искусственный источник s, соединив его с вершинами из V_1, и искусственный сток t, соединив его с вершинами из V_2, добавленным ребрам припишем единичную пропускную способность и нулевую стоимость. Ребрам двудольного графа припишем единичную пропускную способность. Ориентируем все ребра от источника к стоку. Тогда задача о совершенном паросочетании минимального веса сведется к задаче поиска максимального потока минимальной стоимости. Очевидно, что в случае такой специальной структуры графа многие шаги общего алгоритма нахождения потока минимальной стоимости могут быть упрощены.

С другой стороны, задачу о назначениях можно рассматривать как частный случай задачи о наибольшем паросочетании и применить к ней рассмотренный ранее алгоритм, предварительно упростив его. В двудольном графе отсутствуют циклы нечетной длины, поэтому все шаги, связанные со срезанием и распусканием цветком, могут быть опущены.

Получаем следующую задачу линейного программирования P

$$\sum_{j=1}^{q} c_j \xi_j \to \min;$$

$$\sum_{a_j \in T_k}^{q} \xi_j = 1, \quad \forall\, k = 1, \ldots, p, \quad T_k = \{e_j = (v_i, v_k)\};$$

$$\xi_j \geq 0, \quad \forall\, j = 1, \ldots q.$$

Двойственной к этой задаче будет следующая задача P^*

$$\sum_{j=1}^{p} \pi_j \to \max;$$

$$\pi_i + \pi_k \leq c_j, \quad \forall\, j = 1, \ldots, q.$$

Венгерский алгоритм.

Начало. Вершинам из множеств V_1 и V_2 присваиваются веса π_i^1 и π_k^2 так, чтобы для любого ребра $e_j = (v_i, v_k)$ выполнялось $\pi_i^1 + \pi_k^2 \leq c_j$.

Шаг 1. Формируем специальный остовный подграф G' графа G, содержащий только те ребра $e_j = (v_i, v_k)$, для которых $\pi_i + \pi_k = c_j$.

Шаг 2. Если в графе G' нет чередующегося дерева T, то строим новое дерево T с корнем в некоторой открытой вершине $v \in V_1$. Если открытых вершин нет, идем на конец. Если дерево уже есть, продолжаем его построение. Если получено увеличивающее дерево, идем на шаг 3, если венгерское дерево, идем на шаг 4.

Шаг 3. Исключаем из паросочетания M ребра, вошедшие в найденную увеличивающую цепь $< v, y >$, и добавляем к нему оставшиеся ребра этой цепи. Отменяем пометки вершин и окраску ребер дерева. Идем на шаг 2.

Шаг 4. Для ребер $e_j = (v_i, v_k)$, не принадлежащих G', одна концевая вершина которых принадлежит текущему чередующемуся дереву и помечена как внешняя, а другая концевая вершина не принадлежит дереву, находим

$$\Delta = \min_{a_j} [c_j - \pi_i^1 - \pi_k^2].$$

Шаг 5. Для каждой вершины $v_i \in V_1$, являющейся внешней вершиной дерева, увеличиваем π_i^1 на Δ ($\pi_i^1 = \pi_i^1 + \Delta$). Для каждой вершины $v_k \in V_2$, являющейся внутренней вершиной дерева, уменьшаем π_k^2 на Δ ($\pi_k^2 = \pi_k^2 - \Delta$). Сохраняем дерево T и идем на шаг 1.

Конец. Текущее совершенное паросочетание является минимальным.

Матричная форма алгоритма. Операции вышеизложенного алгоритма реализуются на матрице $C : p \times p$, строкам которой соответствуют вершины из V_1, столбцам – вершины из V_2. Первоначально элементы c_{ik} – веса ребер (v_i, v_k).

В начале работы алгоритма находится минимальный элемент в каждой строке и вычитается из всех элементов, и переменная π_i^1 принимает значение этого минимума. Затем та же операция выполняется для столбцов матрицы, и переменная π_k^2 принимает значение минимального элемента в столбце. В результате получается *приведенная матрица* C', для которой выполняется условие

$$c'_{ik} = c_{ik} - \pi_i^1 - \pi_k^2 \geq 0.$$

Граф G' содержит ребра, для которых $c'_{ik} = 0$. Совершенное паросочетание графа G' будет соответствовать набору нулевых элементов в матрице C', для которого в каждой строке и в каждом столбце содержится ровно один нулекой элемент.

Построение чередующегося дерева происходит с помощью расставления пометок в строках и столбцах матрицы C'. Начнем построение произвольного паросочетания в G' окрашивая нулевые элементы матрицы так, чтобы не было двух окрашенных элементов в одной строке либо столбце. Ребра, соответствующие окрашенным нулям, входят в текущее паросочетание, соответствующие неокрашенным нулям – нет.

Найдем строку s без окрашенных элементов и припишем этой строке пометку 0. Если такой строки нет, то текущее паросочетание будет совершенным паросочетанием минимального веса. Иначе вершина v_s является открытой и выбирается в качестве корня чередующегося дерева. Далее выполняем следующие шаги:

а) приписываем пометку i каждому непомеченному столбцу, содержащему неокрашенный ноль в помеченной строке i (если существует несколько возможностей, выбрать любую);

б) приписываем пометку k каждой непомеченной строке, содержащей окрашенный ноль в помеченном столбце k.

Эти операции, выполняемые циклически, строят дерево в G'. Внешние вершины дерева соответствуют строкам, внутренние – столбцам. Если в результате будет помечен столбец, не содержащий окрашенных нулей, то этот столбец соответствует второй открытой вершине, а значит, дерево стало увеличивающим. В нем находится увеличивающая цепь, состоящая из нулей:

$$(s, t_1)(t_1, t_2) \dots (t_{m-1}, t_m),$$

в ней окрашенные нули становятся неокрашенными, и наоборот. Пометки стираются, и процесс продолжается.

Если же расстановку пометок столбцов нельзя будет продолжать (нет непомеченных столбцов, содержащих неокрашенные нули в помеченных строках), дерево станет венгерским. Пусть I^+ и K^+ – множества всех помеченных строк и столбцов соответственно, а I^- и K^- – множества всех непомеченных строк и столбцов, находим

$$\Delta = \min_{i \in I^+, k \in K^-} c'_{ik}.$$

Далее полагаем:

- $\pi_i^1 = \pi_i^1 + \Delta$ для $i \in I^+$;
- $\pi_k^2 = \pi_k^2 - \Delta$ для $k \in K^+$;
- $c'_{ik} = c'_{ik} - \Delta$ для $i \in I^+$ и $k \in K^-$;
- $c'_{ik} = c'_{ik} + \Delta$ для $i \in I^-$ и $k \in K^+$.

Остальные c'_{ik} не меняются. Далее операции а), б) применяются к новой матрице C'.

Пример. Рассмотрим матрицу стоимостей.

	a	b	c	d	e	f	g	h
α	13	21	20	12	8	26	22	11
β	12	36	25	41	40	11	4	8
γ	35	32	13	36	26	21	13	37
δ	34	54	7	8	12	22	11	40
ε	21	6	45	18	24	34	12	48
ζ	42	19	39	15	14	16	28	46
η	16	34	38	3	34	40	22	24
θ	26	20	5	17	45	31	37	43

Построим матрицу C', слева и вверху выпишем весовые векторы, справа и внизу расставим пометки. Сначала выделим жирным шрифтом выделены паросочетание.

		5	0	0	0	0	2	0	3	
		a	b	c	d	e	f	g	h	
8	α	**0**	13	12	4	0	16	14	0	
4	β	3	32	21	37	36	5	**0**	1	g
13	γ	21	29	**0**	23	13	6	0	21	c
7	δ	22	47	0	1	5	13	4	30	0
6	ε	10	**0**	39	12	18	25	6	39	
14	ζ	23	5	25	1	**0**	0	14	29	
3	η	8	31	35	**0**	31	35	19	18	
5	θ	16	15	0	12	40	24	32	35	
				δ				γ		

В строке δ нет окрашенных нулей, она получает пометку 0. Столбец c содержит неокрашенный ноль в строке δ, и он получает пометку δ. Строка γ имеет окрашенный ноль в столбце c и получает пометку c. Затем столбец g получает пометку γ, а строка β – пометку g. Дальше помечать столбцы нельзя, получено венгерское дерево, причем

- $I^+ = \{\beta, \gamma, \delta\}$;
- $K^+ = \{c, g\}$;
- $I^- = \{\alpha, \varepsilon, \zeta, \eta, \theta\}$;
- $K^- = \{a, b, d, e, f, h\}$.

Получаем $\Delta = 1$ и изменяем вектор весов и матрицу.

		5	0	-1	0	0	2	-1	3	
		a	b	c	d	e	f	g	h	
8	α	**0**	13	13	4	0	16	15	0	
5	β	2	31	21	36	35	4	***0***	*0*	g
14	γ	20	28	***0***	22	12	5	*0*	20	c
8	δ	21	46	*0*	0	4	12	4	29	0
6	ε	10	**0**	40	12	18	25	7	39	
14	ζ	23	5	26	1	**0**	0	15	29	
3	η	8	31	36	**0**	31	35	20	18	g
5	θ	16	15	1	12	40	24	33	35	
				δ	δ			γ	β	

Продолжаем расстановку пометок. Столбец d получает пометку δ, столбец h – пометку β. Затем строка η получает пометку g. Столбец h не содержит окрашенных нулей, и он помечен – найдена увеличивающая цепь, ее элементы выделены в матрице курсивом. Теперь окрашенные нули вдоль этой цепи становятся неокрашенными, и наоборот. Все пометки убираются и начинается вторая итерация алгоритма.

		5	0	-1	0	0	2	-1	3	
		a	b	c	d	e	f	g	h	
8	α	**0**	13	13	4	0	16	15	0	
5	β	2	31	21	36	35	4	*0*	***0***	
14	γ	20	28	*0*	22	12	5	***0***	20	
8	δ	21	46	***0***	0	4	12	4	29	
6	ε	10	**0**	40	12	18	25	7	39	
14	ζ	23	5	26	1	**0**	0	15	29	
3	η	8	31	36	**0**	31	35	20	18	
5	θ	16	15	1	12	40	24	33	35	0

Строка θ помечается символом 0, больше пометок поставить нельзя. Получаем $\Delta = 1$, и новая матрица имеет вид.

		5	0	-1	0	0	2	-1	3	
		a	b	c	d	e	f	g	h	
8	α	**0**	13	13	4	0	16	15	0	
5	β	2	31	21	36	35	4	0	**0**	
14	γ	20	28	0	22	12	5	**0**	20	
8	δ	21	46	**0**	0	4	12	4	29	c
6	ε	10	**0**	40	12	18	25	7	39	
14	ζ	23	5	26	1	**0**	0	15	29	
3	η	8	31	36	**0**	31	35	20	18	d
6	θ	15	14	0	11	39	23	32	34	0
				θ	δ					

Столбец c помечается θ, строка δ – пометку c, столбец d – пометку δ, строка η – пометку d. Больше пометок расставить нельзя. Получаем венгерское дерево, $\Delta = 4$, новая матрица имеет вид.

		5	0	-5	-4	0	2	-1	3	
		a	b	c	d	e	f	g	h	
8	α	**0**	13	17	8	0	16	15	0	
5	β	2	31	25	40	35	4	0	**0**	
14	γ	20	28	4	26	12	5	**0**	20	g
12	δ	17	42	***0***	0	*0*	8	0	25	c
6	ε	10	**0**	44	16	18	25	7	39	
14	ζ	23	5	30	5	***0***	*0*	15	29	e
7	η	4	27	36	**0**	27	31	16	14	d
10	θ	11	10	*0*	11	35	19	28	30	0
				θ	δ	δ	ζ	δ		

Продолжается расстановка пометок. Столбцы e и g получают пометку δ, строка ζ – пометку e, строка γ – пометку g. Затем столбец f получает пометку ζ и не содержит окрашенных нулей, значит, найдена увеличивающая цепь, выделенная на матрице курсивом. Перекрашивая нули, получаем следующую матрицу.

		5	0	-5	-4	0	2	-1	3
		a	b	c	d	e	f	g	h
8	α	**0**	13	17	8	0	16	15	0
5	β	2	31	25	40	35	4	0	**0**
14	γ	20	28	4	26	12	5	**0**	20
12	δ	17	42	0	0	**0**	8	0	25
6	ε	10	**0**	44	16	18	25	7	39
14	ζ	23	5	30	5	0	**0**	15	29
7	η	4	27	36	**0**	27	31	16	14
10	θ	11	10	**0**	11	35	19	28	30

Больше строк без окрашенных нулей нет, получено совершенное паросочетание, которое дает паросочетание минимального веса в исходном графе, вес равен 76 (сумме π_i).

6.6. Транспортная задача

Транспортная задача является обобщением задачи о назначениях и получила свое название из-за следующей интерпретации. Пусть имеется n поставщиков и m потребителей, у поставщика с номером i имеется запас α_i единиц ресурса, у потребителя с номером k имеется потребность в β_k единицах ресурса, причем

$$\sum_{i=1}^{n} \alpha_i = \sum_{k=1}^{m} \beta_k.$$

Требуется найти такую схему транспортировки груза, которая имеет минимальную стоимость, если стоимость транспортировки единицы ресурса от i-го поставщика к k-му потребителю равна c_{ik}. Задача линейного программирования имеет вид.

$$\sum_{i=1}^{n}\sum_{k=1}^{m} c_{ik}x_{ik} \to \min;$$
$$\sum_{i=1}^{n} x_{ik} = \beta_k, \quad \forall\, k = 1, \dots m;$$
$$\sum_{k=1}^{m} x_{ik} = \alpha_i, \quad \forall\, i = 1, \dots n;$$
$$x_{ik} \geq 0, \quad \forall\, i = 1, \dots n, \quad k = 1, \dots, m;$$
$$\sum_{i=1}^{n} \alpha_i = \sum_{k=1}^{m} \beta_k.$$

Венгерский алгоритм.

Шаг 1. Начиная с матрицы стоимостей C построить приведенную матрицу C'.

Шаг 2. Пометить строку s, для которой $\alpha_s \neq 0$, пометкой 0. Если таких строк нет, идем на конец.

Шаг 3. Приписать каждому непомеченному столбцу k, содержащему 0 или окрашенный элемент в помеченной строке i, пометку i (если существует несколько возможностей, выбрать любую). Если помечен столбец t с $\beta_t \neq 0$, перейти к шагу 5.

Шаг 4. Приписать каждой непомеченной строке i, содержащей окрашенный элемент в помеченном столбце k, пометку k. Если таких строк нет, перейти на шаг 7, иначе вернуться на шаг 3.

Шаг 5. Найти увеличивающую цепь P между s и t и вычислить

$$\varepsilon = \min\{\alpha_s, \beta_t, \min_{(i,k)\in P^*} c'_{ik}\},$$

где P^* – множество окрашенных ребер в P.

Шаг 6. Попеременно прибавлять $+\varepsilon$ и $-\varepsilon$ ко всем элементам матрица C' в увеличивающей цепи. Окрасить увеличенные элементы и отменить окраску элементов, которые стали равными нулю. Уменьшить α_s и β_t на ε, отменить пометки строк и столбцов и вернуться на шаг 2.

Шаг 7. Пусть I^+ и K^+ – множества всех помеченных строк и столбцов соответственно, а I^- и K^- – множества всех непомеченных строк и столбцов, найти

$$\Delta = \min_{i\in I^+, k\in K^-} c'_{ik}.$$

Далее изменить матрицу C'.:

- $c'_{ik} = c'_{ik} - \Delta$ для $i \in I^+$ и $k \in K^-$;
- $c'_{ik} = c'_{ik} + \Delta$ для $i \in I^-$ и $k \in K^+$.

Остальные c'_{ik} не менять. Сохранить пометки строк и столбцов и перейти на шаг 3.

Конец. Найдено оптимальное решение. Грузы транспортируются только по окрашенным ребрам, c'_{ik} дает величину груза, транспортируемого по ребру (v_i, v_k).

Отметим, что в этом алгоритме матрица C' служит двум целям. Неотмеченые элементы дают предварительные стоимости. Все отмеченные элементы имеют нулевые стоимости, но места, где они расположены, используются на самом деле для хранения величин грузо, транспортируемых по соответствующим ребрам. Это избавляет от необходимости использования отдельной матрицы для хранения этих величин.

Пример. Матрица стоимостей имеет вид

		a	b	c	d	e
		2	*3*	*3*	*5*	*2*
α	*5*	5	3	4	5	6
β	*4*	2	6	5	3	2
γ	*6*	6	4	3	4	4

Приведенная матрица имеет вид (веса вершин здесь не нужны)

		a	b	c	d	e
		2	*3*	*3*	*5*	*2*
α	*5*	2	0	1	1	3
β	*4*	0	4	3	0	0
γ	*6*	3	1	0	0	1

Расставим пометки

		a	b	c	d	e	
		2	*3*	*3*	*5*	*2*	
α	*5*	2	0	1	1	3	0
β	*4*	0	4	3	0	0	
γ	*6*	3	1	0	0	1	
			α				

Найдена увеличивающая цепь – αb, вычисляем $\varepsilon = \min(5, 3) = 3$, изменим матрицу и расставим пометки:

		a	b	c	d	e	
		2	*0*	*3*	*5*	*2*	
α	*2*	2	0/3*	1	1	3	0
β	*4*	0	4	3	0	0	
γ	*6*	3	1	0	0	1	
			α				

Получено венгерское дерево, меняем матрицу. Здесь $I^+ = \{\alpha\}$, $K^+ = \{b\}$, $\Delta = 1$. Продолжаем расстановку пометок.

		a	b	c	d	e	
		2	*0*	*3*	*5*	*2*	
α	*2*	1	$0/3^*$	0	0	2	0
β	*4*	0	5	3	0	0	
γ	*6*	3	2	0	0	1	
			α	α	α		

Найдена увеличивающая цепь – αc, вычисляем $\varepsilon = \min(2, 3) = 2$, изменим матрицу и расставим пометки:

		a	b	c	d	e	
		2	*0*	*1*	*5*	*2*	
α	*0*	1	$0/3^*$	$0/2^*$	0	2	
β	*4*	0	5	3	0	0	0
γ	*6*	3	2	0	0	1	
		β			β	β	

Найдена увеличивающая цепь – βa, вычисляем $\varepsilon = \min(4, 2) = 2$, изменим матрицу и расставим пометки:

		a	b	c	d	e	
		0	*0*	*1*	*5*	*2*	
α	*0*	1	$0/3^*$	$0/2^*$	0	2	
β	*2*	$0/2^*$	5	3	0	0	0
γ	*6*	3	2	0	0	1	
		β			β	β	

Найдена увеличивающая цепь – βd, вычисляем $\varepsilon = \min(2, 5) = 2$, изменим матрицу и расставим пометки:

		a	b	c	d	e	
		0	*0*	*1*	*3*	*2*	
α	*0*	1	$0/3^*$	$0/2^*$	0	2	
β	*0*	$0/2^*$	5	3	$0/2^*$	0	
γ	*6*	3	2	0	0	1	0
				γ	γ		

Найдена увеличивающая цепь – γc, вычисляем $\varepsilon = \min(6, 1) = 1$, изменим матрицу и расставим пометки:

		a	b	c	d	e	
		0	*0*	*0*	*3*	*2*	
α	*0*	1	$0/3^*$	$0/2^*$	0	2	
β	*0*	$0/2^*$	5	3	$0/2^*$	0	
γ	*5*	3	2	$0/1^*$	0	1	0
				γ	γ		

Найдена увеличивающая цепь – γd, вычисляем $\varepsilon = \min(5, 3) = 3$, изменим матрицу и расставим пометки:

		a	b	c	d	e	
		0	*0*	*0*	*0*	*2*	
α	*0*	1	$0/3^*$	$0/2^*$	0	2	
β	*0*	$0/2^*$	5	3	$0/2^*$	0	d
γ	*2*	3	2	$0/1^*$	$0/3^*$	1	0
		β		γ	γ	β	

Найдена увеличивающая цепь – $\gamma d \beta e$, вычисляем $\varepsilon = \min(2, 2, 2^*) = 2$, изменим матрицу и расставим пометки:

		a	b	c	d	e	
		0	*0*	*0*	*0*	*0*	
α	*0*	1	$0/3^*$	$0/2^*$	0	2	
β	*0*	$0/2^*$	5	3	0	$0/2^*$	d
γ	*0*	3	2	$0/1^*$	$0/5^*$	1	0
		β		γ	γ	β	

Варианты транспортной задачи.

Задача на избыток. Запасы поставщиков превышают потребности потребителей, т. е.

$$\sum_{i=1}^{n} \alpha_i > \sum_{k=1}^{m} \beta_k.$$

Вводится стоимость хранения поставщиком i единицы товара d_i, т. е. убытки, которые несет поставщик, если не реализует товар. Тогда задача выглядит так

$$\sum_{i=1}^{n}\sum_{k=1}^{m} c_{ik}x_{ik} + \sum_{i=1}^{n}\left(d_i - \sum_{k=1}^{m} x_{ik}\right) \to \min;$$

$$\sum_{i=1}^{n} x_{ik} = \beta_k, \quad \forall\, k = 1, \dots m;$$

$$\sum_{k=1}^{m} x_{ik} \le \alpha_i, \quad \forall\, i = 1, \dots n;$$

$$x_{ik} \ge 0, \quad \forall\, i = 1, \dots n, \quad k = 1, \dots, m;$$

$$\sum_{i=1}^{n} \alpha_i = \sum_{k=1}^{m} \beta_k.$$

Данная задача сводится к исходной введением фиктивного потребителя с номером 0 и стоимостями перевозок $c_{i0} = d_i$.

Задача на недостаток. Запасы поставщиков меньше, чем потребности потребителей, т. е.

$$\sum_{i=1}^{n} \alpha_i < \sum_{k=1}^{m} \beta_k.$$

Аналогично, вводится фиктивный поставщик с номером 0 и стоимостями перевозок $c_{0k} = d_k$, где d_k – потери, которые несет потребитель из-за недопоставки товара.

Задача с запретами. Если существует запрет на поставки от i-го поставщика к j-му потребителю, то задача сводится к предыдущей, если положить для таких пар стоимость перевозок $c_{ik} = M > \max\{c_{ik}\} \sum\limits_{i=1}^{n} \alpha_i$. При этом если решение задачи включает такие запрещенные поставки, то система ограничений несовместна.